R E P U B L I Q U E F R A N Ç A I S E

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 30 MAI 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

CERTIFIED COPY OF
PRIORITY DOCUMENT

1er depot

INPI

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Remplir impérativement la 2ème page.

Cet imprimé est à remplir lisiblement à l'encre noire DB 540 W / 190600

Réservé à l'INPI

REMISE DES PIÈCES
DATE 5 JUIL. 2002
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0203491

DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 05 JUIL. 2002

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

CABINET NETTER
36 avenue Hoche
75008 PARIS

Vos références pour ce dossier *(facultatif)* SUN 38 (120710)

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	Cochez l'une des 4 cases suivantes
Demande de brevet	☒
Demande de certificat d'utilité	☐
Demande divisionnaire	☐
Demande de brevet initiale	N° Date / /
ou demande de certificat d'utilité initiale	N° Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N° Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Extending role scope in a directory server system.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	Pays ou organisation Date / / N°
	Pays ou organisation Date / / N°
	Pays ou organisation Date / / N°
	☐ S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»

5 DEMANDEUR		☐ S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»
Nom ou dénomination sociale		SUN MICROSYSTEMS, INC
Prénoms		
Forme juridique		
N° SIREN		. . . . .
Code APE-NAF		. . . .
Adresse	Rue	901 San Antonio Road
	Code postal et ville	94303 PALO ALTO Californie
Pays		Etats-Unis d'Amérique
Nationalité		Société des Etats-Unis d'Amérique
N° de téléphone *(facultatif)*		
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		

1er dépôt

INPI

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DES PIÈCES DATE 5 JUIL 2002

LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0208491

DB 540 W / 190600

Vos références pour ce dossier : *(facultatif)*		SUN 38 (120710)
6 MANDATAIRE		
Nom		PLAÇAIS
Prénom		Jean-Yves
Cabinet ou Société		Cabinet NETTER
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	36 avenue Hoche
	Code postal et ville	75008 Paris
N° de téléphone *(facultatif)*		01 58 36 44 22
N° de télécopie *(facultatif)*		01 42 25 00 45
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		
Les inventeurs sont les demandeurs		☐ Oui ☒ Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☐ ☒
Paiement échelonné de la redevance		**Paiement en deux versements, uniquement pour les personnes physiques** ☐ Oui ☐ Non
9 RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence) :*
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)

N° Conseil 92-1197 (B) (M)
Jean-Yves PLAÇAIS

VISA DE LA PRÉFECTURE OU DE L'INPI

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

Extending role scope in a directory server system

This invention relates to distributed computer systems.

In certain fields of technology, complete computer systems, including a diversity of equipments, are optimized for storing and retrieving data. Such systems may provide services to user machines related to a local network, e.g., an Intranet, or to a global network, e.g., the Web network.

It is desirable that network users can access, upon a query, to a large number of data, making it possible for the network users to create their own dynamic web site or to consult a dynamic web site, for example an e-commerce site on a multi platform computer system (Solaris, Windows NT). These queries are directed to a directory, e.g., a LDAP directory, and managed by a directory server. It is further desirable that this access to a large number of data be made possible more rapidly for each query arriving after a first query.

A general aim of the present invention is to provide advances in these directions.

Broadly, there is proposed a method of operating a directory server system, comprising a directory server interacting with entries organized in a tree structure. The entries comprise user entries and role entries. Each role entry defines a role, and has an associated scope in the tree, the scope being defined from the location of the role entry in the tree, according to a predefined rule. The role of an existing role entry is attached to a user entry subject to a first condition, which comprises a role membership condition and the fact that the user entry belongs to the scope of that existing role entry. The method comprises the steps of:

a) adding extra role data to a given role entry, the extra data identifying an extra scope in the tree for that given role entry,
b) attaching the role of that given role entry to a user entry subject to a second condition, which comprises said role membership condition and the fact the user entry belongs to the extra scope of that given role entry.

There is also proposed a directory server system, comprising:

- a directory server interacting with entries organized in a tree structure. The entries comprise user entries and role entries. Each role entry defines a role, and has an associated scope in the tree, the scope being defined from the location of the role entry in the tree, according to a predefined rule,
- a role mechanism, capable of attaching the role of an existing role entry to a user entry, subject to a first condition, which comprises a role membership condition and the fact that the user entry belongs to the scope of that existing role entry,
- the role mechanism is further capable of determining whether an existing role entry has extra data designating an extra scope, and, if so, of attaching the role of that role entry to a user entry subject to a second condition, which comprises said role membership condition and the fact the user entry belongs to the extra scope of that given role entry.

This invention may also be defined as an apparatus or system, and/or as software code for implementing the method, or for use in the system, and/or as portions of such software code, in all their alternative embodiments to be described hereinafter.

Other alternative features and advantages of the invention will appear in the detailed description below and in the appended drawings, in which :

- Figure 1 is a general diagram of a computer system in which the invention is applicable;
- Figure 2 illustrates a typical LDAP exchange between a LDAP client and a LDAP server, and between the LDAP server and additional servers;
- Figure 3 represents the general structure of a LDAP directory;
- Figure 4 shows an exemplary portion of a LDAP tree;
- Figure 5 represents attribute types and values of an entry;
- Figure 6 is a portion of a LDAP tree showing the scope of a role, according to the prior art;
- Figure 7 is a table showing three types of LDAP roles according to the prior art;
- Figures 8a to 8c represent the structure of the three types of roles shown in figure 7;
- Figure 9 shows a portion of a LDAP tree structure, according to the prior art;
- Figure 10 is a general flowchart for determining whether a given entry is member of an existing role, in accordance with an embodiment of this invention;
- Figure 11 is a table, showing syntax of an extended role, in accordance with the invention;

– Figure 12 is a flowchart for determining whether a given entry is member of an existing role, in accordance with an alternative embodiment of this invention;
– Figure 13a is a flowchart for enumerating the roles of a given entry according to the prior art;
– Figure 13b is a flowchart for enumerating the roles of a given entry according to an embodiment of this invention;
– Figure 14 represents portions of a LDAP tree comprising extending roles in accordance with an embodiment of this invention;
– Figure 15 to 17 and 18a to 18c represent the structure of role caches associated with top suffixes according to the invention ; and
– Figure 19 shows a tree structure portion having an extending role.

Additionally, the detailed description is supplemented with the following Exhibits:
– Exhibit E1 contains examples of roles according to the prior art ;
– Exhibit E2 contains an example of an extending role in accordance with the invention.

In the foregoing description, references to the Exhibits are made directly by the Exhibit or Exhibit section identifier: for example, E2.1 refers to section E2.1 in Exhibit E2. The Exhibits are placed apart for the purpose of clarifying the detailed description, and of enabling easier reference.

Now, making reference to software entities imposes certain conventions in notation. Particularly, an expression indicated between the quote signs " " may be used to design LDIF extracts and an expression in italics may be used for representing an attribute and an object class.

As they may be cited in this specification, Sun, Sun Microsystems and Sun One are trademarks of Sun Microsystems, Inc.

Trademark Office patent file or records, but otherwise reserves all copyright and/or author's rights whatsoever.

This invention may be implemented in a computer system, or in a network comprising computer systems. Figure 1 represents an example of the hardware of such computer systems. The hardware comprises :

- a processor CPU 11, e.g. an Ultra-Sparc;
- a program memory 12, e.g. an EPROM, a RAM, or Flash memory;
- a working memory 13, e.g. a RAM of any suitable technology;
- a mass memory 14, e.g. one or more hard disks;
- a display 15, e.g. a monitor;
- a user input device 15, e.g. a keyboard and/or a mouse;
- a network interface device 21 connected to a communication medium 20, which is in communication with other computers. Network interface device 21 may be of the type of Ethernet, or of the type of ATM . Medium 20 may be based on wire cables, fiber optics, or radio-communications, for example.

Data may be exchanged between the components of figure 1 through a bus system 10, represented as a single bus for simplification of the drawing. Bus systems may include a processor bus, e.g. PCI, connected via appropriate bridges to, e.g. an ISA or an SCSI bus.

The data exchanged are handled by a resource provider using a server to deliver data to user computers, or to store the data provided by the user computers. Browsers, e.g. Internet Explorer, are further provided on user computers, to enable users, to make requests to retrieve or store data. The resource provider makes it possible for user computers on a network to share data of any kind.

iPlanet E-commerce Solutions, now Sun One E-commerce Solutions, has developed a "net-enabling" platform called the Internet Service Deployment Plateform (ISDP). ISDP includes multiple, integrated layers of software that provide a full set of services supporting application development, e.g., business-to-business exchanges, communications and entertainment vehicles, and retail Web sites.

Sun One™ Directory Server provides a centralized directory service directory service for an intranet or an extranet. A directory service represents a collection of software, hardware, and processes that are able to deliver and store information. The directory service generally includes one or more directory client programs that can access the data stored in the directory, e.g. names, phone numbers or addresses.

The Sun One Directory Server is a general purpose directory that stores all information in a single, network-accessible repository. The Sun One Directory Server provides the standard protocol LDAP and an application programming interface (API) to access the information contained by the Sun One Directory Server.

LDAP is the Internet standard for directory lookups, just as the Simple Mail Transfer Protocol (SMTP) is the Internet Standard for delivering e-mail and the Hypertext Transfer Protocol (HTTP) is the Internet standard for delivering documents. Technically, LDAP is defined as on-the-wire bit protocol (similar to HTTP) that runs over Transmission Control Protocol/ Internet Protocol (TCP/IP). It specifies the interaction between clients and servers and determines how LDAP queries and responses are carried over the IP network.

A LDAP-compliant directory, such as the Sun One Directory Server, leverages a single, master directory that contains all users, groups and access information. The directory is hierarchical, not relational and is particularly fitted for reading while offering a high reliability and a high scalability.

For example, the directory can be used to provide information technology managers with a list of all the hardware and software assets in a widely spanning enterprise. Most importantly, a directory server provides resources that all applications can use, and aids in the integration of these applications that have previously functioned as stand-alone systems. Instead of creating an account for each user in each system the user needs to access, a single directory entry is created for the user in the LDAP directory.

Referring now to figure 2, LDAP defines a communication 1 between a server 17 and a client 18. LDAP also defines a communication 2 between LDAP server 17 and servers 17.1 to 17.n, which makes it possible for the server LDAP 17 to exchange its content (replication

service) with servers 17.1 to 17.n or to access the directory of one of the servers 17.1 to 17.n (referral service) and vice versa.

The LDAP protocol is a message-oriented protocol. The client 18 constructs a LDAP message containing a request and sends the message to the server 17. The server 17 processes the request and sends a result, or results, back to the client 18 as a series of LDAP messages.

Such a client-server communication additionally lies on a specific architecture. LDAP creates a standard defining the way data are exchanged between the client computer and the directory server and defining the way data are modeled. More specifically, LDAP relies on four basic models:

- an information model;
- a naming model;
- a functional model; and
- a security model.

The LDAP information model defines the kind of data that can be stored in a directory. LDAP directory is populated with entries. An entry corresponds to real-world objects, such as a person, a printer, or configuration parameters.

Figure 3 illustrates the general structure of a LDAP directory : the directory server 30 executes implemented functions based on the entries 31 stored in databases. The entries comprise configuration entries 310, user entries 311 and administrative entries 312. These entries further interact with the schema 32 which is described below.

The configuration entries are stored under the subtree "cn=config". The user entries comprise data related to the users of the directory server. Administrative entries relate to user management and are generally implemented as LDAP subentries.

An entry contains a set of attributes associated with values (e.g. common name *cn* or surname *sn*). Each entry is uniquely identified by a distinguished name DN. This distinguished name may be stored in the attribute *dn* (distinguishedName) of each entry.

LDAP entries are organized in a hierarchical tree structure, called the Directory Information Tree (DIT). Each node of the tree comprises an entry. Figure 4 illustrates an organization entry (90) with the attribute type of domain component *dc*, an organizational unit entry (92) with the attribute type of organizational unit *ou*, a server application entry (94) with the attribute type of common name *cn*, and a person entry (96) with the attribute type of user ID *uid*. The entries are connected by the directory. Each server has a particular entry called root directory specific entry (rootDSE) which contains the description of the tree and of its content.

A LDAP Data Interchange Format (LDIF) is an ASCII text file format used to describe directory entries and operations on those entries. It enables to create, modify, and delete Directory entries and to import and export data among LDAP directories. Figure 5 is a LDIF representation of an entry 404, showing the attribute types 400 and their values 402.

The information model is extensible, which means that new types of information can be added to a LDAP directory.

Descriptive information is stored in the attributes of the entry. Each attribute describes a specific type of information. Attributes may have constraints that limit the type and length of data placed in attribute values.

All entries require the *objectclass* attribute which lists the object classes to which an entry belongs. An entry can belong to one or more object classes and must satisfy all of them. The *objectclass* attribute defines which attributes are required and which attributes are allowed in the entry.

For example, in figure 5, the entry (404) represented in LDIF belongs to the object classes *top*, *person*, *organizationalPerson* and *inetOrgPerson*.

Each attribute has a corresponding syntax definition. The syntax definition describes the type of information provided by the attribute. The object classes, the required and allowed attributes, and the syntax definition of the attributes are listed in the directory schema.

The LDAP directory comprises a structure 32, represented in figure 3, that defines object classes and attributes, and may be viewed as metadata. This structure, called the schema, sets the rules defining what information can be stored in the LDAP directory and how information is organized. The schema specifies the required and allowed attributes that are used to store information and their syntax definition. A schema checking function may be activated, thus causing the directory server to check new entries to verify whether :

- object classes and attributes attached to new entries are defined in the schema 32,
- the attributes required for an object class according to the schema 32, are contained in an entry attached to that object class,
- only attributes allowed by the object class according to the schema 32, are contained in an entry attached to that object class.

The LDAP naming model specifies that directory entries must be hierarchical and organized in an inverted tree structure. As mentioned above, each entry has a unique name called a distinguished name *dn*. The *dn* consists of a list of the names of all the parent entries in the directory back to the top of the directory hierarchy, the name of the entry being at the extreme left, e.g., "uid=Joe,ou=people,dc=france,dc=sun,dc=com", in figure 5. The root of the entry is at the extreme right of the *dn*. The name at the extreme left of the *dn*, "uid=Joe" in the example, is the relative distinguished name or *rdn*. Within a set of entries sharing the same parents, the *rdn* must be unique. This ensures that two entries in the directory tree cannot have the same *dn*.

The LDAP functional model comprises eight basic functional operations (indicated thereinafter between the quote signs " ") that a user from a client computer can perform on the directory data of a LDAP directory server :

- "bind" and "unbind" : begin and end the exchange of information between LDAP clients and the directory server;
- "add", "delete", and "modify" : apply on specific entries in the DIT,
- "compare" : applies on two entries to compare their content according to criteria,
- "search" : locates specific entries in the DIT,
- "modifyRDN" : applies to change the distinguished name *dn* of an entry.

In addition to the eight basic functional operations, the LDAP protocol defines a framework for adding new operations to the protocol via LDAP extended operations. Extended operations allow the protocol to be extended in an orderly manner to meet new marketplace needs as they emerge.

According to another aspect of LDAP directories, entry grouping mechanisms are provided to simplify the management of LDAP users. Roles, been introduced in the version 5 of iDS, constitute a LDAP grouping mechanism.

A role may have members, which are the entries said to possess the role. Role mechanism enables the following operations:

- enumerating the members of a given role,
- determining whether a given entry possesses a particular role,
- enumerating all the roles possessed by a given entry,

Additionally, it is possible to assign a particular role to a given entry and to revoke a particular role from a given entry.

Every role is defined by its own definition entry. A role is uniquely identified by the distinguished name of its definition entry. Role definition entries are LDAP subentries and therefore inherit the subentry mechanism, defined in the ISO/IEC X.509 standard, for scoping. The scope of a role corresponds to the subtree of the role parent entry as illustrated by figure 6. User entries E01 and E02 are in the scope S1 of the role R1 while entry E11 is out of the scope of the role R1. Thus, E01 and E02 are likely to be members of role R1 while E11 cannot be member of role R1.

Referring to figure 7, a role can be of "managed" type 701, "filtered" type 702 or "nested" type 703. Each type of role further has two specific object classes 71 that inherit from the *nsRoleDefinition* object class, and is related to specific attributes 72 (*nsRoleDN*, *nsRoleFilter*).

On creating a role, members may be assigned to the role as follows:

- members of a managed role have the *nsRoleDN* attribute in their entry,

- members of a filtered role are entries that match the filter specified in the *nsRoleFilter* attribute of the role definition entry,
- members of a nested role are members of the roles specified in the *nsRoleDN* attributes of the nested role definition entry.

Managed roles belong to the *nsSimpleRoleDefinition* object class and are thus simple roles. Filtered and nested roles belong to the *nsComplexRoleDefinition* object class are thus complex roles.

Exhibits E1.1, E1.3 and E1.5 respectively represent a managed role, a filtered role and a nested role, in LDIF, according to the prior art. Exhibits E1.2 and E1.4 represent respectively a user entry, member of the managed role of exhibit E1.1, and a user entry, member of the filtered role of exhibit E1.3.

Figures 8a, 8b and 8c respectively illustrate a managed role, filtered or nested role. As represented in figure 8a, *nsRoleDN* relates a user Entry E0 to a managed role (*ManagedRole*), thus indicating that the entry is member of that role. Figure 8b shows that *nsRoleFilter* relates a filtered role (*FilteredRole*) to a user entry E0, thus indicating that entry E0 is member of that role. *nsRoleDN* can also be a pointer from a nested role to another role (*ManagedRole*, *FilteredRole*) as shown in figure 8c.

LDAP servers uses a computed attribute called *nsrole* to determine the roles possessed by a given entry. The *nsrole* attribute is a computed attribute and as a result it is not stored in the entry itself. This attribute is computed for each of the three types of role operations identified above.

In figure 9, a user 134 identified by the RDN "cn = rob" in the engineering division 131 ("o=eng") is member of the engineering role 133 ("cn = engrole"). He has the permissions associated with that role. However, he may need to access a resource, e.g. an application, requiring the permissions associated with sales role 135 ("cn = salesrole") defined for the sales division 132 o = sales. In other words, this implies that the user "cn = rob" needs to be added as member of the sales role 135, while remaining member of the engineering role 133.

As the scope of the sales role 135 is defined by the subtree 1302, a solution of the prior art is to add an additional entry "cn = rob", having an attribute "nsRoleDN = salerole" as a subentry of the entry 136 "ou = people" of sales division 132. However, this solution is not easy to apply when the sales division and the engineering division are managed by two distinct role administrators, as often happens. Moreover, it allows the user to have only the permissions authorized for the sales role or only the permissions authorized for the engineering role, but not permissions authorized for both. This solution is thus limited.

Another known solution is to create a managed role "cn=ManagedEngRole" possessing the user-oriented entry "cn=bob" and a nested role that contains both the sales role 132 and the role possessing the user "cn=bob". This requires the new nested role entry to be located at the top level of the directory tree structure, which is extremely complicated to administer.

The invention addresses the above problem.

The invention implements a new method for extending role scope.

The invention proposes to extend the scope of role entries based on extra data. According to the invention, the extra data identify an extra scope of the DIT and may be added to role entries to extend their scope. The extra data are enabled in the LDAP metadata, as required.

According to an aspect of the invention, the extra data may comprise a special attribute *nsRoleScopeDN* that identifies the extra scope.

For each one of these roles, if the corresponding role entry does not comprise the special attribute *nsRoleScopeDN*, the scope of the role is the scope defined by the *LDAPsubentry* object class, in accordance with the prior art.

If the corresponding role entry comprises the special attribute *nsRoleScopeDN*, the scope of the role entry is the union of the scope defined through the *LDAPsubentry* object class and of the extra scope indicated by the special attribute. Thus, a user entry or a group of user entries being in an extra scope can be member of a given role, even if the extra scope is different from the scope of the given role. Consequently, a user can benefit from a

permission restricted to a particular role, even if he is not in the organization for which the role is defined.

Reference is now made to figure 10, illustrating the different operations performed for checking whether a given user entry E0 is member of a given role R1, according to the invention. As in the prior art, the directory tree comprises role entries. However in accordance with the invention, some of the role entries may comprise the special attribute *nsRoleScopeDN*.

At the initial operation 100, the directory server receives the request. Operation 101 retrieves role data associated with role R1. These role data are represented by a data structure comprising specific information about the role, like the type of the role, e.g. "nested", the filter condition when the role is filtered and the role distinguished name *dn*. The role data may be stored in a cache to ease the processing of the request. They are provided from the attributes comprised in the role definition entry.

Operation 102 determines if entry E0 is in the scope of role R1. This operation may be performed, comparing part of the distinguished names of entry E0 and role R1.

If entry E0 is in the scope of role R1, operation 103 further checks whether role R1 is nested.

If role R1 is not nested, the server tests, at operation 104, if entry E0 matches the membership condition of role R1 :
- If role R1 is filtered, the membership condition corresponds to the filter condition as is provided by the role data;
- If role R1 is managed, the membership condition is related to the role *dn* provided by the role data,

If entry E0 matches the membership condition of role R1, operation 106 determines that entry E0 is member of role R1.

If role R1 is nested, at operation 105, the directory server recursively performs operations 102 and 104 for each role contained by the nested role, and therefore the membership

condition corresponds to the membership conditions of the roles contained by the nested role.

If entry E0 is in the role scope of one of the roles contained by role R1 and if entry E0 matches the membership condition of that role, i.e if entry E0 is member of one the role contained by role R1 (test 107), operation 109 determines that entry E0 is member of role R1.

At operation 102, if entry E0 is not in the scope of role R1, operation 108 determines whether role R1 has the special attribute *nsRoleScopeDN*. If not, operation 116 determines that entry E0 is not member of role R1.

If role R1 has the special attribute *nsRoleScopeDN*, this special attribute identifying an extra scope, operation 110 checks if entry E0 is in the extra scope, and if so, operation 103, described above, is performed.

In accordance with this embodiment of this invention, the special attribute *nsRoleScopeDN* designates the Distinguished Name of the extra scope.

According to an alternative embodiment of this invention, further to the extra scope, the extra data of a given role may identify an appended role being out of the scope of the role. The extra scope is thus a scope of the tree structure that includes the scope of the role being appended. The extra scope may be for example the scope of the appended role. In particular, the extra data may comprise the special attribute identifying the extra scope and another attribute identifying the appended role.

According to this embodiment of the invention, the directory server enables an entry or a set of entries being in an extra scope to be member of a role, if the following conditions are fulfilled:

- the entry or set of entries is/are member of an appended role;
- the role has the extra data;
- the appended role and the extra scope are identified by the extra data of the role.

It is no more required that the entry or set of entries belong to the scope of the role entry, like in the prior art.

With reference to figure 9, e.g., the entry "cn=rob" belonging to the appended role "cn=engrole" could be member of the role "cn=salesrole", if the role entry "cn=salesrole" is added the extra data and if the extra data identify the appended role "cn=engrole" and an extra scope including the scope 1300.

In accordance with this embodiment, the special attribute, called *nsRoleScopeDN* is attached to nested roles and designates the DN of a location in the tree, the extra scope being the subtree of this location.

Nested role entries are provided to contain one or more roles of any type, these roles being in the scope of the nested role entry. The attribute *nsRoleDN* of the nested role defines the distinguished name *dn* of the contained roles. The structure of nested roles has been improved to also enable an appended role being out of the scope of a nested role entry to be contained by the nested role entry, and as a result to enable a group of entries out of the scope of the nested role to be member of the nested role. To perform such an extension, the value of the *nsRoleScopeDN* attribute designates the location of an extra scope that includes the appended role scope, and a value of the *nsRoleDN* attribute identifies the appended role.

Consequently, the directory server may use the *nsRoleDN* attribute designating the DN of an appended role and the *nsRoleScopeDN* attribute designating an extra scope to enable an entry possessing the designated appended role to further possess the nested role.

The table of Figure 11 illustrates the structure of a nested role having an extending scope. A nested role entry according to the prior art is identified by the attribute *dn*, and belongs to the object classes *LDAPsubentry*, *nsRoleDefinition*, *nsComplexRoleDefinition* and *nsNestedRoleDefinition*. Moreover, a role contained by the nested role is identified by the attribute *nsRoleDn*. According to the prior art, the values of this attribute designate the distinguished names (*dn*) of the contained roles. The syntax of these distinguished names may directly give information about the scope of the contained roles. According to an aspect

of the invention the distinguished names of the contained roles may indicate that their scope is different from the scope of the nested role.

The scope extension of nested roles is made possible by the *nsRoleScopeDn* attribute. This attribute is defined in LDAP schema, and may be associated with the *nsNestedRoleDefinition* object class. Consequently, any nested role entry of the directory tree may be added the *nsRoleScopeDn* attribute. The syntax of this attribute is a distinguished name syntax.

Additionally, the role data associated with nested roles may comprise the extra scope or extended scope designated by *nsRoleScopeDN*. The extended scope may be updated in the role data, in response to the *nsRoleScopeDN* attribute being modified, added or deleted.

Figure 12 is a flowchart showing alternatives to operations 108 and 110 of figure 10, according to this embodiment.

If operation 102 of figure 10 determines that entry E0 is not in the scope of role R1, operation 1080 of figure 12 checks whether role R1 is nested. If role R1 is nested, operation 1081 determines if role R1 further has an extended scope based on the role data and if so, operation 1100 determines whether entry E0 is in the extended scope. In response to entry E0 being in the extending scope, operation 1101 determine whether entry E0 is member of an appended role contained by the nested role. Preferably, this operation is performed by determining whether entry E0 is member of one of the roles contained by nested role R1, according to operations 105 and 107 of figure 10. If test 1101 succeeds, operation 1090 determines that entry E0 is member of role R1. If one of the previous tests fails, operations 1160 determines that entry E0 is not member of role R1.

Alternatively, it is possible that several appended roles may be contained by a nested role, provided that all those appended roles be in the extra scope

An important function related to roles is to enumerate all the roles possessed by a user. The concept of extending scope for roles involves modifications to that function. The embodiment based on nested roles makes it possible to use parts of the existing operations used for enumerating the roles possessed by a user.

Figure 13a is a flowchart representing the different operations implemented to perform such a function according to the prior art and figure 13b is a flowchart representing the different operations implemented to perform such a function according to the invention.

In response to a request for enumerating all the roles possessed by a given user entry E0, in operation 200, the directory server proceeds to the computing of *nsrole* attribute for that entry. *nsrole* attribute is a multi-valued attribute indicating all the roles possessed by a user entry.

The existing function for computing *nsrole* is based on testing whether the given entry is member of a set of candidate roles. For each one of the candidate roles, the directory server tests if the given entry E0 is member of that role, as explained above, in reference to the flowcharts of figures 10 and 12.

In the prior art, when a request is made to determine the roles possessed by a given user entry, the set of candidate roles is determined from a list of roles associated with the top suffix of the given entry. This list is prepared in advance in a role cache. The role cache is a data structure updated on creating a new role or on deleting an existing role in the subtree of the top suffix. The role cache contains the list of the roles defined in the subtree of the top suffix. Each role of the role cache is also related to the role data.

Figure 13a illustrates the processing of a request for listing the roles possessed by a given user entry E0 :

- at operation 200, the directory server receives the request for listing the roles possessed by a given user entry E0;
- the computing of *nsrole* attribute starts with operation 202, that performs access to the top suffix of entry E0;
- operation 204 retrieves the role cache associated with the top suffix of E;
- for each role of the cache role,
  - operation 206 retrieves the role data of the current role;
  - operation 208 tests if entry E0 is member of the current role, and if so adds the role to the result and select the next role of the list;

- when all the candidate roles have been tested (operation 210 fails), the directory server assigns the result to *nsrole*, at operation 212.

According to another aspect of this invention, the directory tree structure may comprise extended roles i.e roles having extended scope. Consequently, a given entry E0 may be member either of one of the roles associated with its top suffix, or of one of the roles of another top suffix.

Figure 14 illustrates parts of a directory tree structure, comprising an extended role. Exhibit E2 comprises the definition of this structure in LDIF. In the prior art, to find all the roles possessed by the given entry 246 "cn=bob,o=user,o=qa,o=suffix2", the directory server tests the candidates roles of its top suffix "o=suffix2". This test indicates that entry 246 is member of the role 243, "cn=marketing". But, the entry 246 is also member of an extended role 142 "cn=everybody_cross2" under another top suffix 140 "o=suffix1", which is not found by the function of the prior art.

The invention proposes to modify the selection of the candidate roles in order to further find the extended roles possessed by a given entry.

For each top suffix of the directory tree structure, for each role of the role cache associated with that suffix, the directory server may test if entry E0 is member of that role according to the operations of the figure 10 or 12. However, as a directory tree structure may comprise a great number of role entries, this may not be optimum.

Alternatively, the following operations may be performed :

a1) executing the function for listing all the role possessed by entry E0, according to the prior art, as illustrated in figure 13a. This operation provides the roles possessed by entry E0, among the roles associated with its top suffix, these roles not being extended;

b1) for each top suffix of the directory tree structure, testing if entry E0 is member of one of the extended roles among the candidates roles of that suffix, according to the flowchart of figures 10 and 12.

An embodiment of operation b1) is illustrated by the flowchart of figure 13b. To determine the extended roles associated with a top suffix of the tree structure (request 500), operation 500 retrieves the role cache of the current top suffix. In accordance with an aspect of this embodiment, the role cache further comprises a field indicating the list of extensions of the top suffix, the extensions representing the distinguished names of the extended scopes defined in nested roles under the top suffix. This list is prepared in advance from the role data of those nested roles.

After retrieving the list of extensions (operation 502), operation 503 determines, for each extension, whether the given entry E0 is in the scope defined by the extension. This may be done by comparing the distinguished name of the entry with the distinguished name of the extension.

If entry E0 is in the scope defined by the extension, operation 504 gets the nested roles associated with the current extension. Operation 506 then tests if entry E0 is member of each one of these roles, according to the operations of the flowchart represented in figure 10 or 12. When entry E0 is determined to be member of a role, the role is added to the result (operation 507)

After checking all the roles associated with the current extension (operation 508), the directory server recursively repeats operations 503 to 508 for all the extensions. When all the extensions have been checked (test 509), the directory server repeats operations 501 to 509 for the next top suffix of the tree structure. When all the suffixes have been checked (test 510), the directory server assigns the result to the *nsrole* attribute of entry E0 (operation 511).

Reference is now made to figures 15a to 15c, showing an exemplary structure of a role cache. This structure known in the art, has been modified to enable the extension of role scope. A role cache maintains the following information for a given top suffix, as shown in figure 15a:

– the DN of the top suffix in field L1,

– the list of the roles associated with that top suffix in field L2,

In accordance with an aspect of this invention, the role cache also provides a pointer L3 to the list of the top suffix extensions, defined in nested roles under this given top suffix. This pointer indicates the DN of an extension L31 of the top suffix and a pointer L32 to the next extension.

More specifically, the list of the roles may be represented with binary search trees, like AVL trees (named after their inventors Adelson-Velskii-Landis). The structure of figure 15a illustrates the structure of an AVL tree. An AVL tree representing a role comprises a pointer to the root of the AVL tree, in field L2.

Each role is represented by an AVL tree. An AVL node of an AVL tree may be represented by the structure of figure 15b, comprising a pointer on the left AVL node L21, a pointer on the right AVL node L23, and AVL data L22 corresponding to the role data. When the role is nested, the right or the left AVL node represents the distinguished names of the roles contained by the nested role.

Referring to figure 17, the AVL data represent the role data. In particular, they comprise the following information :

- the distinguished name of the role entry L221,
- the role type L222 (nested /managed/filtered).

If the role is filtered, the AVL data further comprise the filter condition defined by the role filter.

According to another aspect of the invention, the AVL data may comprise the DN of the extended scope in field L223. This information is related to the value of *nsRoleScope*DN attribute in the role entry and correspond to one of the extension of the suffix.

Figures 16 and 18a to 18c illustrate an example of a role cache of an extended role, in relation with the role entries defined in exhibits E2.13, E2.14 and E2.18. The field L2 of the role cache of root suffix "o=suffix1" is a pointer on the AVL tree represented in figure 16.

The AVL node representing the role "cn=everybody_cross,o=suffix1" of field L22.1, has a pointer L23.1 on the AVL tree a first contained role L22.2 "cn=staff, o=qa,o=suffix_2". Moreover, the AVL node representing this first contained role, has a pointer L21.2 on the AVL tree of a second contained role L22.3 "cn=staff, o=qa,o=suffix_1".

Reference is now made to figures 18a to 18c, representing the AVL data of the roles L22.1, L22.2 and L22.3, referred above. The field L222.1 of figure 18a indicates that the role L22.1 "cn=everybody_cross,o=suffix1" is nested. The AVL data of that role also comprise the field L223 comprising a distinguished name indicating an extended scope. The extension of the scope of the role L22.1 is performed using the information contained in fields L22.2 and L223.

Figures 18b and 18c represent the AVL data of the roles contained by the role "everybody_cross, o=suffix1". The field L221.2 of figure 18b indicates that the role having the DN "cn=staff,o=qa,o=suffix2" stored in field L221.2 is managed. The field L221.3 of figure 18c indicates that the role having the DN "cn=staff,o=qa,o=suffix12" stored in field L221.2 is also managed.

On attaching the *nsRoleScopeDN* to a nested role, the directory server proceeds to the following update operations :

i1) updating the field L223 "scope extension DN" in the role data associated with the nested role;

ii1) updating the field L22 "extensions" L31 in the role cache of the nested role suffix.

In accordance with another embodiment of this invention, it is possible to enable a single user in a first scope to be member of a nested role having a second scope, the first scope and the second scope being different. This function is performed by the following operations:

i2) creating an appended role, preferably a managed role, that only possesses the single user;

ii2) adding the attribute *nsRoleScopeDN*, invoking the extra scope and the attribute *nsRoleDN*, invoking the *dn* of the appended role to selected nested roles of the tree structure, in order to enable the single user to be member of those roles.

Referring now to figure 9, many users, not represented, may be member of the role "cn=engrole". To enable the single user "cn=rob" of the suffix 134 "o=eng" to be member of the role 135 "cn=salesrole", according to operation i2), a managed role 137 "cn=managedEngRole" has been created. The corresponding structure is represented in figure 19. According to operation ii2), the role 135 "cn=salesrole" is added :

- the attribute *nsRoleScopeDN*, invoking the distinguished name of the extra scope;
- the attribute *nsRoleDN*, invoking the distinguished name of the role managedEngRole "o=eng,dc=France,dc=sun,dc=com".

In the prior art, the enumeration of the entries possessed by a given role is performed by computing the *nsrole* attribute of candidate entries. The operations for computing *nsrole* attribute according to the invention, have been described in reference to figure 13a.

This invention also encompasses software code, especially when made available on any appropriate computer-readable medium. The expression "computer-readable medium" includes a storage medium such as magnetic or optic, as well as a transmission medium such as a digital or analog signal. Such software code may include data and/or metadata.

This invention also encompasses the software code to be added to existing directory server functionalities to perform anyone of the various new functionalities, as described above, which may be used independently of each other.

On another hand, a number of features have been positively described, using absolute language, to help understanding the LDAP example. Each such feature should be considered as exemplary only, and is not intended to restrict the scope of this invention in any way.

Exhibit E1

E1.1 - Managed role

```
dn : cn = Marketing, ou = people, dc = example, dc = com
objectclass : top
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsSimpleRoleDefinition
objectclass : nsComplexRoleDefinition
cn : Marketing
description : managed role for marketing staff
```

E1.2 - Entry member of Marketing role

```
dn : cn = Joe, ou = people, dc = example, dc = com
objectclass : person
cn : Joe
sn : Bradford
userpassword : joepasswd
nsRoleDN : cn = Marketing, ou = people, dc = example, dc = com
```

E1.3- Filtered role

```
dn : cn = SalesFilter, ou = people, dc = example, dc = com
objectclass : top
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsComplexRoleDefinition
objectclass : nsFilteredRoleDefinition
cn : SalesFilter
nsRoleFilter : description=marketing guy
description : filtered role for sales staff
```

E1.4- Entry member of Filtered role

```
dn : cn = Richard, ou = people, dc = example, dc = com
objectclass : person
cn : Richard
sn : Parker
userpassword : richardpasswd
description : marketing guy
```

E1.5- Nested role

```
dn : cn = MarketingSales, ou = people, dc = example, dc = com
objectclass : top
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsComplexRoleDefinition
objectclass : nsNestedRoleDefinition
cn : MarketingSales
nsRoleDN : cn = Marketing, ou = people, dc = example, dc = com
nsRoleDN : cn = SalesFilter, ou = people, dc = example, dc = com
description : nested role for marketing and sales staff
```

Exhibit E2 : Example of extended role

E2.1- suffix definition 1

```
dn : o=suffix_1
objectclass : organization
objectclass : top
```

E2.2- organization entry

```
dn : o=qa,o=suffix_1
objectclass : organization
objectclass : top
```

E2.3- user organization unit entry

```
dn : ou=users,o=qa,o=suffix_1
objectclass : organizationalUnit
objectclass : top
```

E2.4- extended role entry

```
dn : cn=everybody_cross2,o=qa,o=suffix_1
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsComplexRoleDefinition
objectclass : nsNestedRoleDefinition
nsRoleScopeDN : o=suffix_2
nsroledn : cn=marketing,o=qa,o=suffix_2
nsroledn : cn=staff,o=qa,o=suffix_1
```

E2.5- staff role

```
dn : cn=staff,o=qa,o=suffix_1
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsSimpleRoleDefinition
objectclass : nsManagedRoleDefinition
```

E2.6- suffix definition 2

```
dn : o=suffix_2
objectclass : organization
objectclass : top
```

E2.7- organization entry

```
dn : o=qa,o=suffix_2
objectclass : organization
objectclass : top
```

E2.8- user organization unit entry

```
dn : ou=users,o=qa,o=suffix_2
objectclass : organizationalUnit
objectclass : top
```

E2.9- marketing role

```
dn : cn=marketing,o=qa,o=suffix_2
objectclass : LDAPsubentry
objectclass : nsRoleDefinition
objectclass : nsSimpleRoleDefinition
objectclass : nsManagedRoleDefinition
```

E2.10- user entry

```
dn : cn=bob,ou=users,o=qa,o=suffix_2
objectclass : top
objectclass : person
sn : kap
userpassword : secret
nsroledn : cn=marketing,o=qa,o=suffix_2
```

Claims

1. A method of operating a directory server system, comprising a directory server interacting with entries organized in a tree structure,
in which said entries comprise user entries and role entries,
each role entry defining a role, and having an associated scope in the tree, the scope being defined from the location of the role entry in the tree, according to a predefined rule,
with the role of an existing role entry being attached to a user entry subject to a first condition, which comprises a role membership condition and the fact that the user entry belongs to the scope of that existing role entry,
the method comprising the steps of:

a) adding extra role data to a given role entry, the extra data identifying an extra scope in the tree for that given role entry,
b) attaching the role of that given role entry to a user entry subject to a second condition, which comprises said role membership condition and the fact the user entry belongs to the extra scope of that given role entry.

2. The method of claim 1, wherein the existing role entry is an indirect role entry, designating one or more other roles.

3.The method of claim 5, wherein the existing indirect role entry has an attribute designating the said one or more other roles.

4. The method of claim 1 through 3, wherein the role membership condition comprises the fact the user entry has an attribute designating the role in the existing role entry.

5. The method of claim 1 through 3, wherein the existing role entry has a role filter condition, and the role membership condition comprises the fact that one or more attributes of the user entry meet the role filter condition.

6.The method of claim 5, wherein the existing role entry has an attribute designating the role filter condition.

7. The method of claim 1, wherein step a) comprises:

a) adding the extra role data to the given role entry, in the form of an added attribute having a special attribute name, and being associated with an attribute value, identifying the extra scope.

8.The method of claim 7, wherein the value of the special attribute, designates a location in the tree outside of the scope as defined from said predefined rule, and the extra scope is defined from that designated location in accordance with a second predefined rule.

9.The method of claim 8, wherein the extra scope is defined as the subtree of the designated location.

10.The method as claimed in any of the preceding claims, wherein the predefined rule comprises defining the scope of a role entry as the subtree of the parent of that role entry in the tree.

11. The method as claimed in any of the preceding claims, further comprising the step of responding to a request of whether a designated user entry has a given role by:

i1. determining a role entry corresponding to the given role,

i2. determining whether the user entry meets the first condition, and

i3. if not, determining whether the role entry has extra data identifying an extra scope, and, if so, determining whether the user entry meets the second condition.

12. The method as claimed in any of the preceding claims, further comprising the step of responding to a request for user entries having a given role by:

j1. determining a role entry corresponding to the given role,

j2. scanning the tree to determine user entries meeting the first condition,

j3. determining whether the role entry has extra data identifying an extra scope, and,

j4. if so, scanning the tree to determine user entries meeting the second condition.

13. The method as claimed in any of the preceding claims, further comprising the step of responding to a request for the roles of a given user entry by:

k1. determining a role entry,

k2. determining whether the given user entry meets the first condition for that role entry,

k3. if not, determining whether the role entry has extra data identifying an extra scope, and, if so, determining whether the given user entry meets the second condition for that role entry,

k4. repeating steps k1. through k3. with other roles entries until an end condition is met.

14. The method of claim 13, wherein the end condition comprises having scanned substantially all the applicable roles.

15. The method as claimed in claim 13, in which the given entry belongs to the subtree of a top suffix of the tree structure, wherein

- step k2. is performed for each role belonging to the subtree of said top suffix, and
- step k3. is performed for each top suffix of the tree structure, for each role belonging to the subtree of said top suffix.

16. A directory server system, comprising:

- a directory server interacting with entries organized in a tree structure, said entries comprise user entries and role entries, each role entry defining a role, and having an associated scope in the tree, the scope being defined from the location of the role entry in the tree, according to a predefined rule,
- a role mechanism, capable of attaching the role of an existing role entry to a user entry, subject to a first condition, which comprises a role membership condition and the fact that the user entry belongs to the scope of that existing role entry,
- said role mechanism being further capable of determining whether an existing role entry has extra data designating an extra scope, and, if so, of attaching the role of that role entry to a user entry subject to a second condition, which comprises said role membership condition and the fact the user entry belongs to the extra scope of that given role entry.

17.The directory server system of claim 16, in which the existing role entry is an indirect role entry, designating one or more other roles.

18.The directory server system of claim 17, in which the existing indirect role entry has an attribute designating the said one or more other roles.

19. The directory server system of claim 16 through 18, wherein the role membership condition comprises the fact the user entry has an attribute designating the role in the existing role entry.

20. The directory server system of claim 16 through 18, wherein the existing role entry has a role filter condition, and the role membership condition comprises the fact one or more attributes of the user entry meet the role filter condition.

21.The directory server system of claim 20, wherein the existing role entry has an attribute designating the role filter condition.

22. The directory server system of claim 16, wherein the extra data of the role entry comprise an added attribute having a special attribute name, and being associated with an attribute value, identifying the extra scope.

23.The directory server system of claim 22, wherein the value of the special attribute, designates a location in the tree outside of the scope as defined from said predefined rule, and the extra scope is defined from that designated location in accordance with a second predefined rule.

24.The directory server system of claim 23, wherein the extra scope is defined as the subtree of the designated location.

25. The directory server system as claimed in any of claims 16 through 24, wherein the predefined rule comprises defining the scope of a role entry as the subtree of the parent of that role entry in the tree.

26. The directory server system as claimed in any of claims 16 through 25, wherein the role mechanism has a first function for responding to a request of whether a designated user entry has a given role, said first function being capable of:

i1. determining a role entry corresponding to the given role,

i2. determining whether the user entry meets the first condition, and

i3. if not, determining whether the role entry has extra data identifying an extra scope, and, if so, determining whether the user entry meets the second condition.

27. The directory server system as claimed in any of claims 16 through 26, wherein the role mechanism has a second function for responding to a request for user entries having a given role, said second function being capable of:

j1. determining a role entry corresponding to the given role,

j2. scanning the tree to determine user entries meeting the first condition,

j3. determining whether the role entry has extra data identifying an extra scope, and,

j4. if so, scanning the tree to determine user entries meeting the second condition.

28. The directory server system as claimed in any of claims 16 through 27, wherein the role mechanism has a third function for responding to a request for the roles of a given user entry, said third function being capable of:

k1. determining a role entry,

k2. determining whether the given user entry meets the first condition for that role entry,

k3. if not, determining whether the role entry has extra data identifying an extra scope, and, if so, determining whether the given user entry meets the second condition for that role entry,

k4. repeating steps k1. through k3. with other roles entries until an end condition is met.

29.The directory server system of claim 28, wherein the end condition comprises having scanned substantially all the applicable roles.

30. The directory server system of claim 28 or 29, in which the given entry belongs to the subtree of a top suffix of the tree structure, wherein:

- step k2. is performed for each role belonging to the subtree of said top suffix, and
- step k3. is performed for each top suffix of the tree structure, for each role belonging to the subtree of said top suffix.

31. The software code for performing the steps of any of claims 1 through 15.

32. The software code for a directory server in a directory server system as claimed in any of claims 16 through 30.

33. A software code for a directory server having a role mechanism, capable of attaching the role of an existing role entry to a user entry, subject to a first condition, which comprises the fact that the user entry belongs to the scope of that existing role entry,
said software code enhancing the role mechanism to be further capable of determining whether an existing role entry has extra data designating an extra scope, and, if so, of attaching the role of that role entry to a user entry subject to a second condition, which comprises the fact the user entry belongs to the extra scope of that given role entry.

34. The combination of the software code of claim 33 with the role mechanism being enhanced.

CABINET NGUYEN

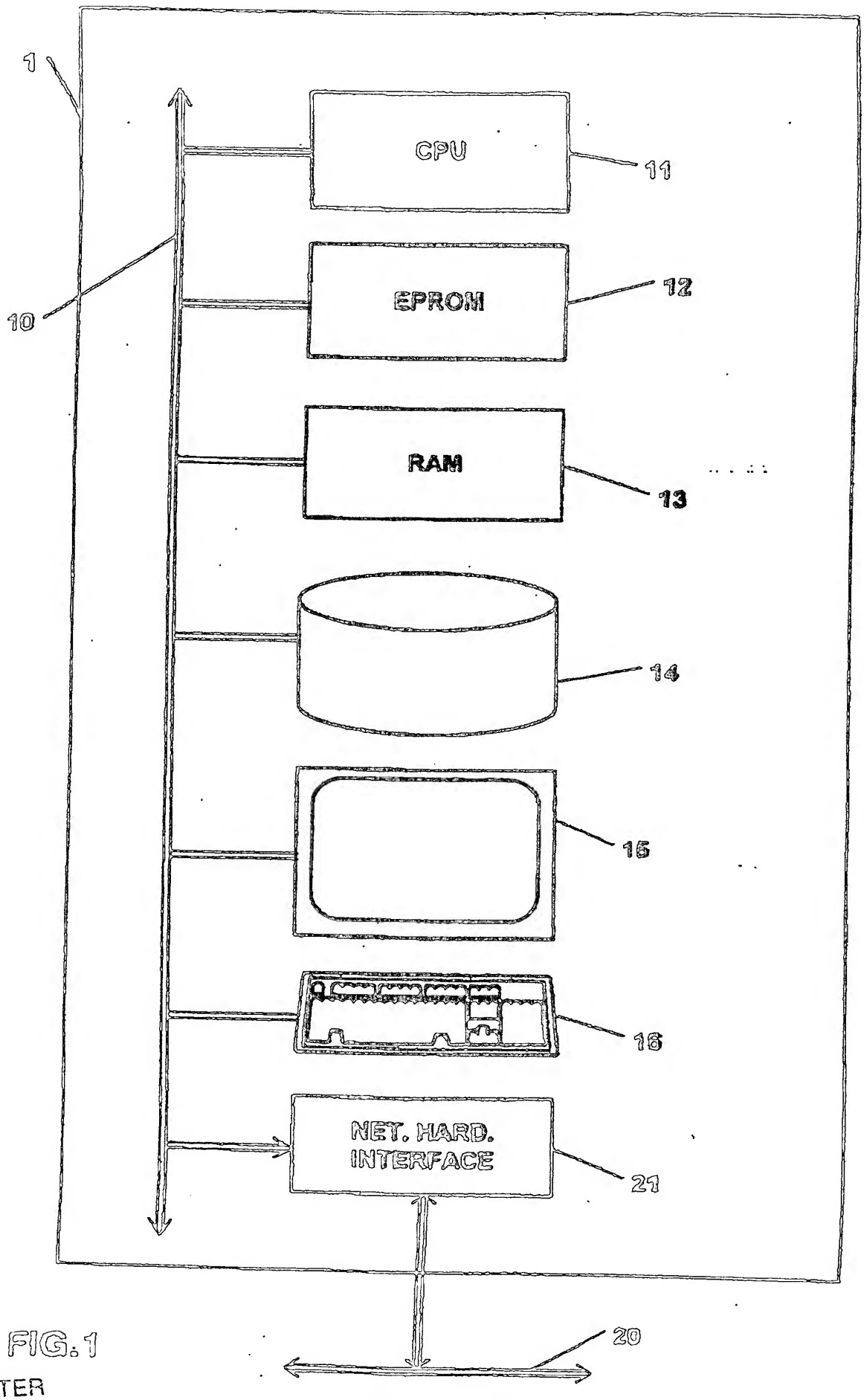


FIG. 1

PRIOR ART

CABINET NETTER

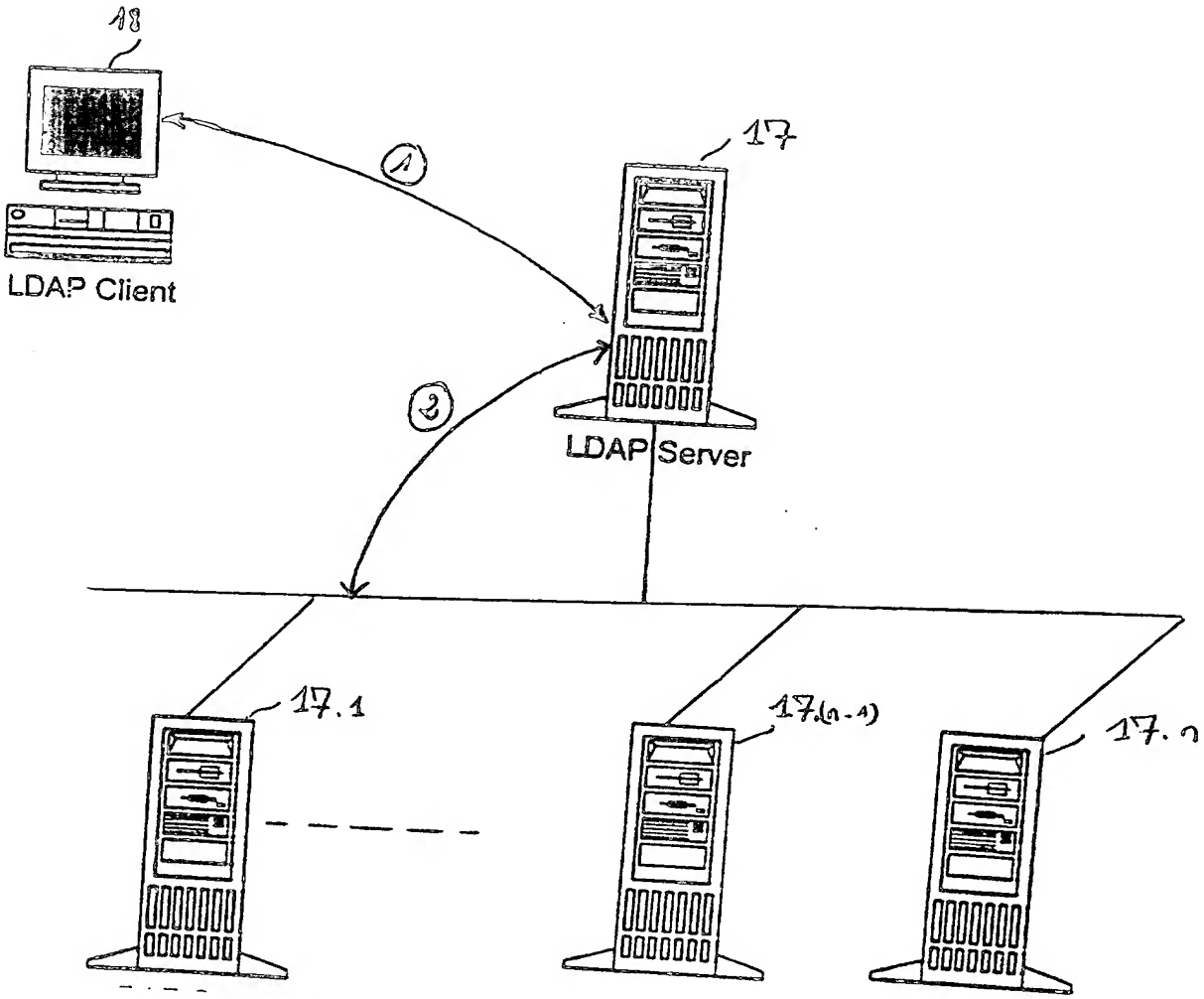


FIG.2.

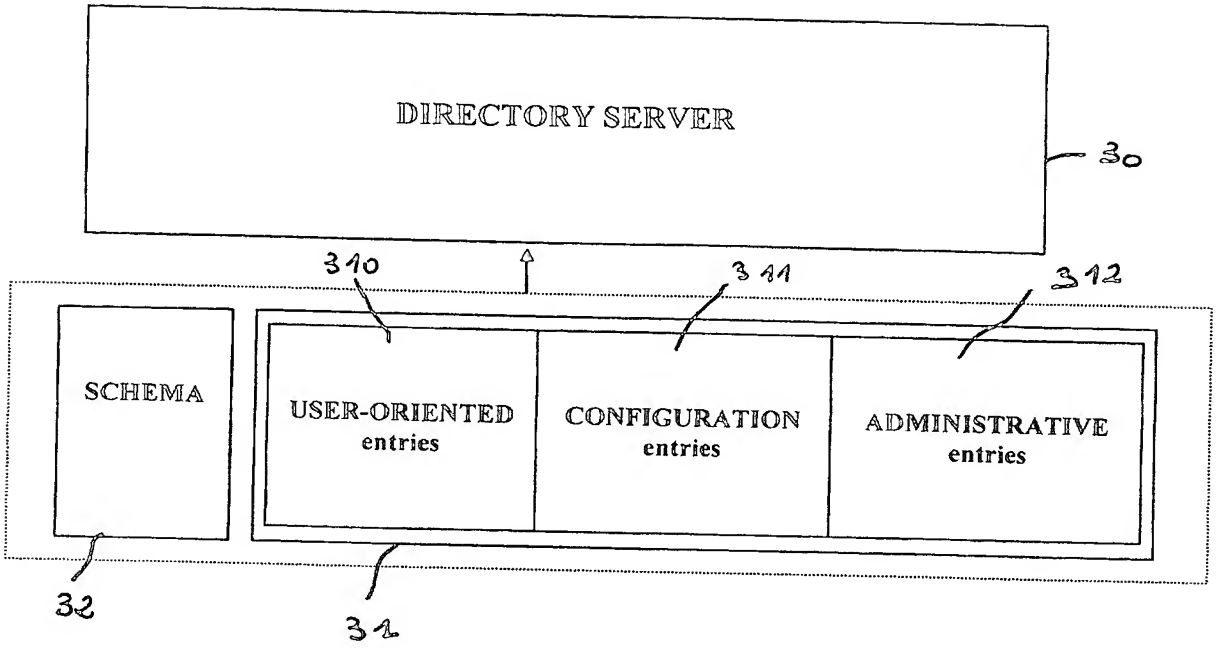


FIG 3

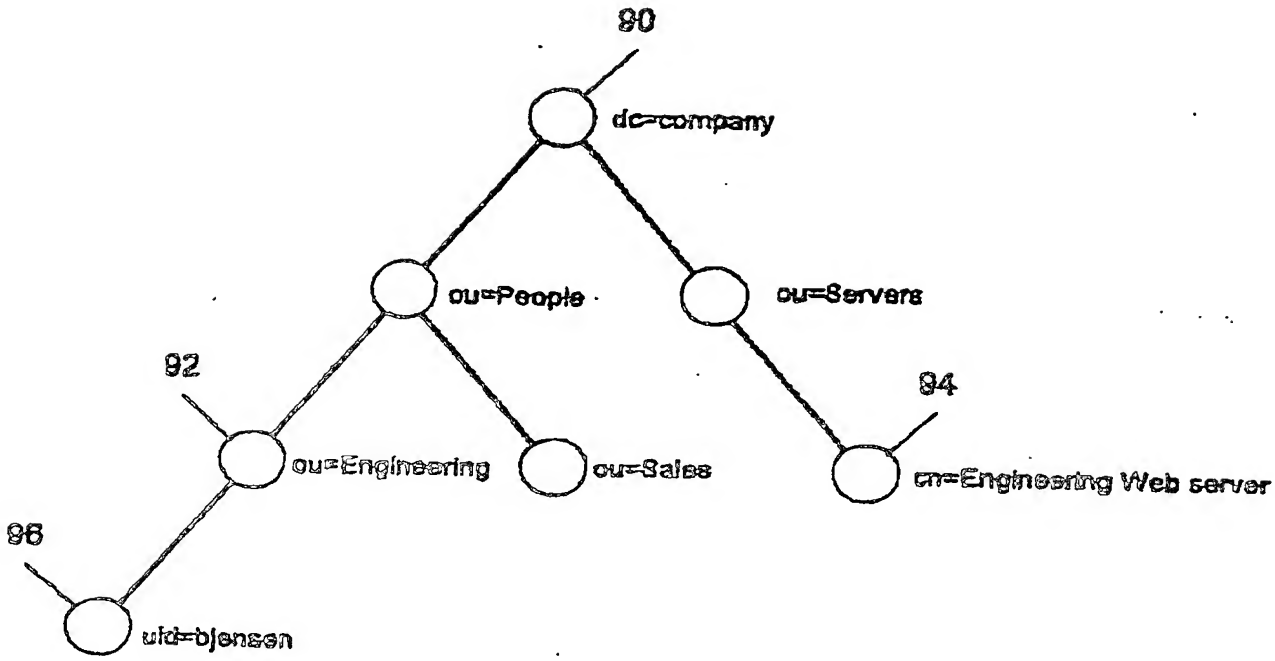


FIG 4

(PRIOR ART)

Entry 404

Attribute type 400	Attribute value 402
dn :	uid=Joe, ou=people, dc=france, dc=sun, dc=com
objectClass :	top
objectClass :	person
objectClass :	organizationalPerson
objectClass :	inetOrgPerson
cn :	Joe
sn :	Rayan
uid :	joerayan
mail :	joerayan@sun.com
phoneNumber :	778

Fig 5

(PRIOR ART)

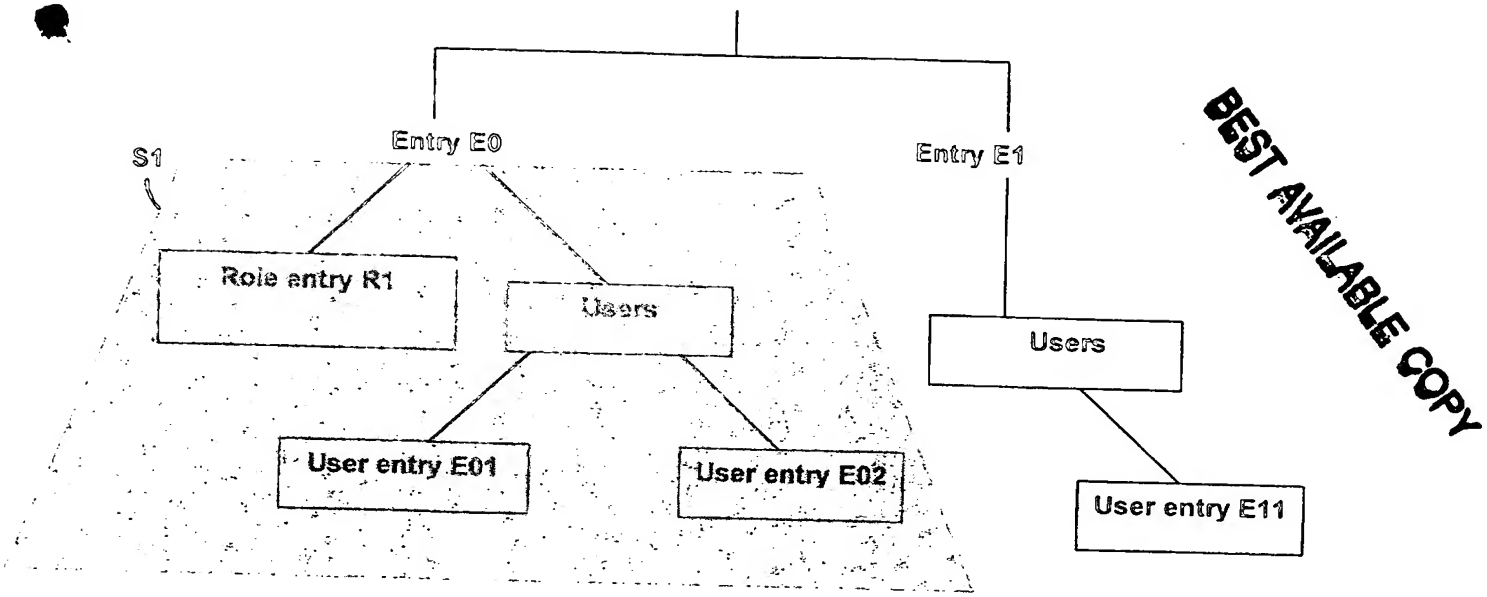


FIG 6

(PRIOR ART)

Role Type	Object Classes	Attributes
Managed Role	nsSimpleRoleDefinition nsManagedRoleDefinition	description (optional)
Filtered Role	nsComplexRoleDefinition nsFilteredRoleDefinition	NsRoleDN description (optional)
Nested Role	nsComplexRoleDefinition nsNestedRoleDefinition	NsRoleDN description (optional)

71 72 701 702 703

Figure 7

(PRIOR ART)

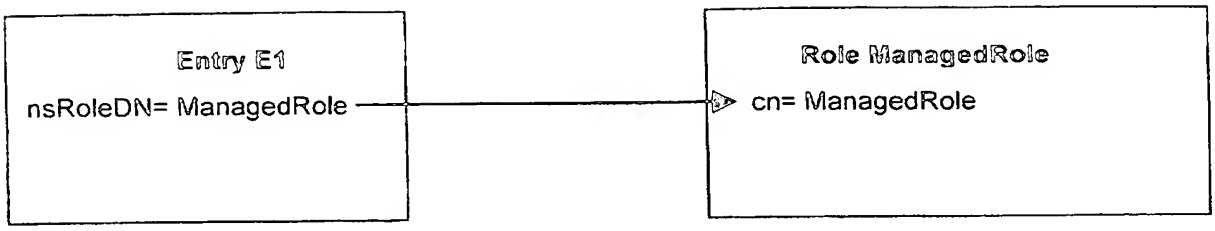


FIG 8a

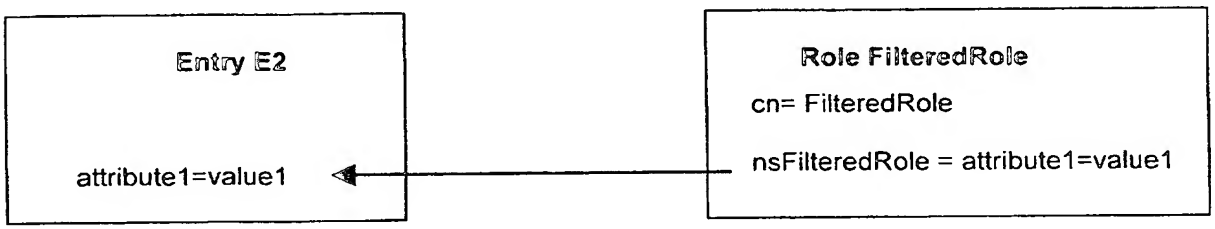


FIG 8b

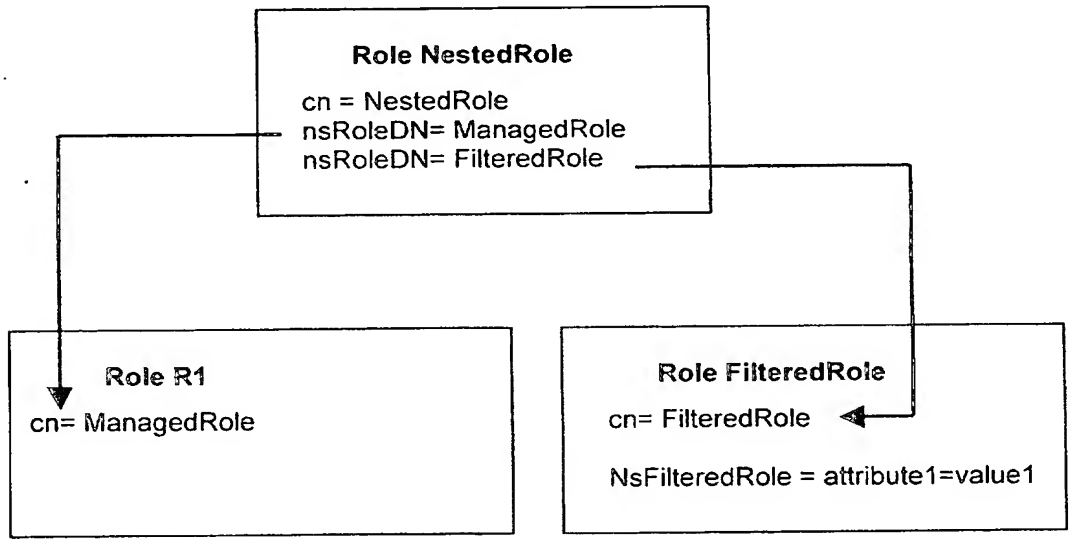


FIG 8c

(PRIOR ART)

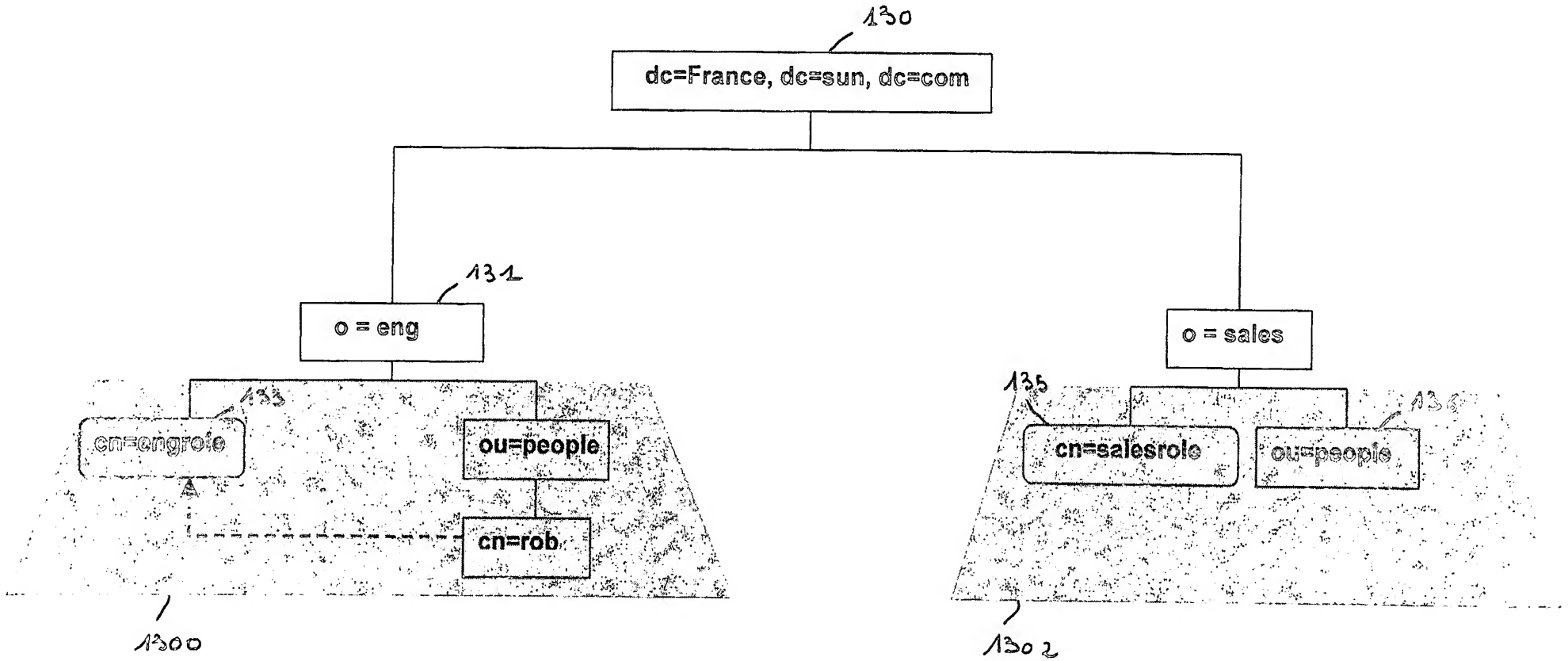


Figure 9

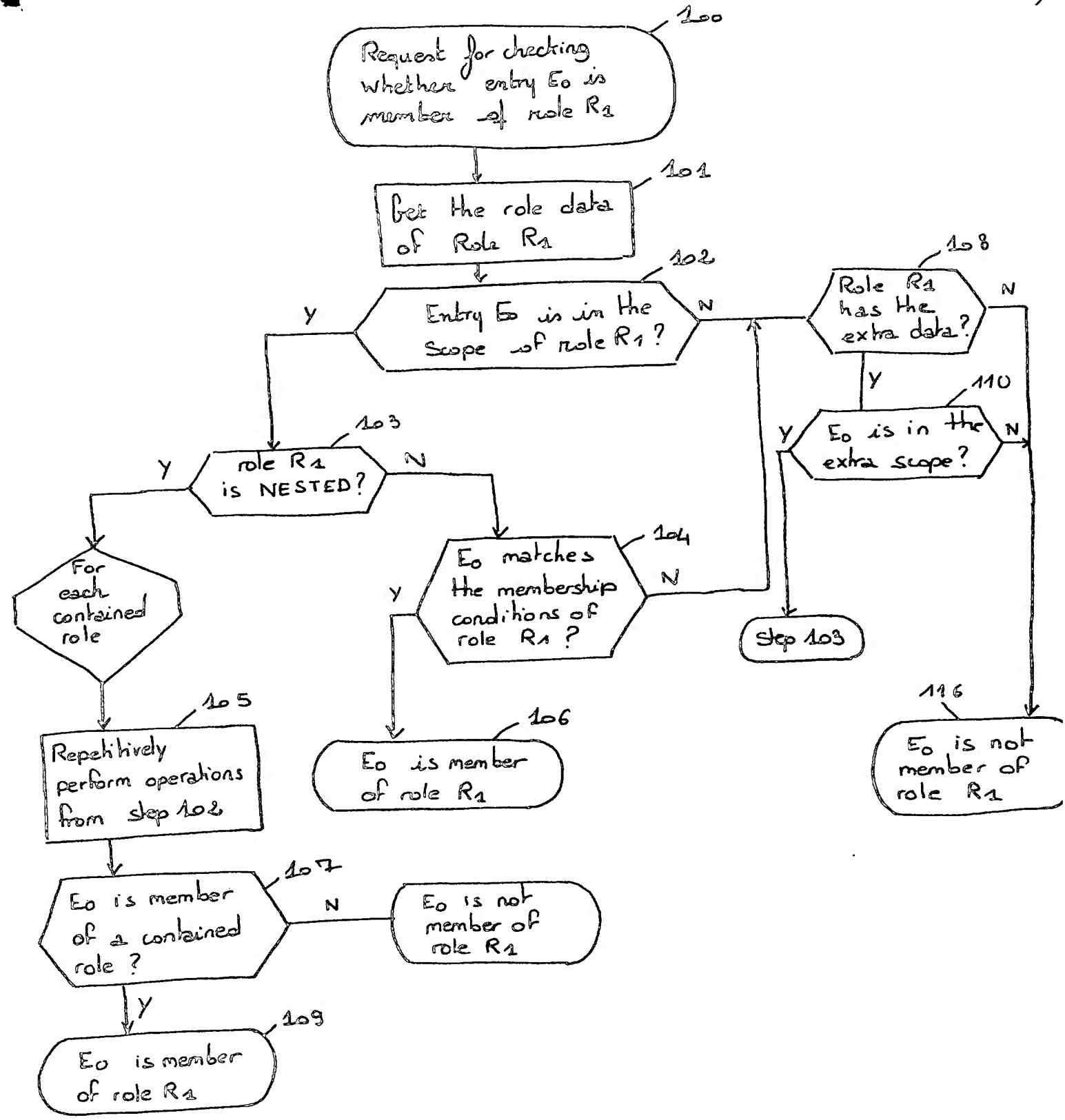


Figure 1a

ATTRIBUTES	ATTRIBUTE VALUES
dn	distinguished name of the nested role
objectclass	object classes of the nested role
nsRoleScopeDn	distinguished name of the extended scope
nsRoleDn	distinguished names of the roles contained by the nested role

Figure 11

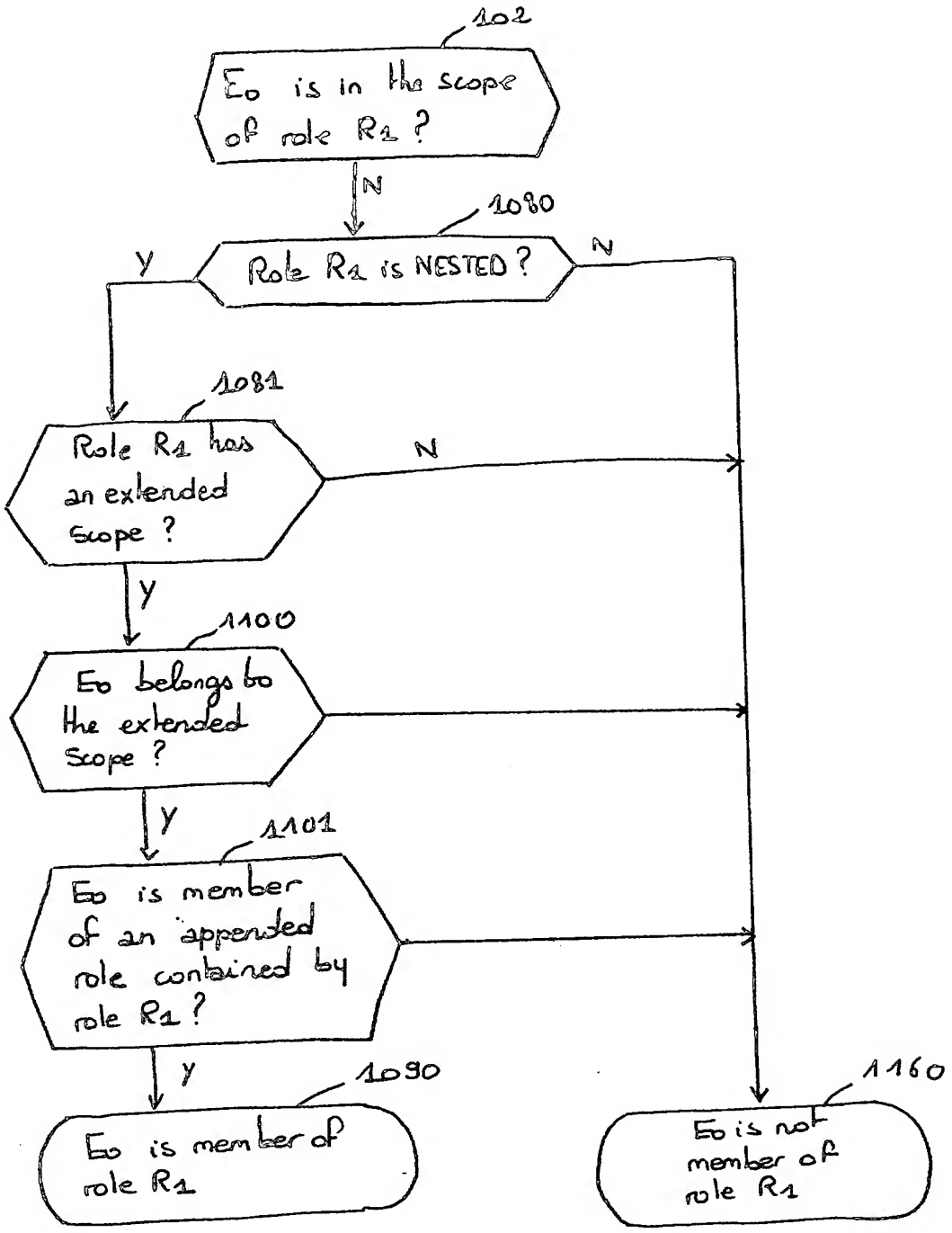
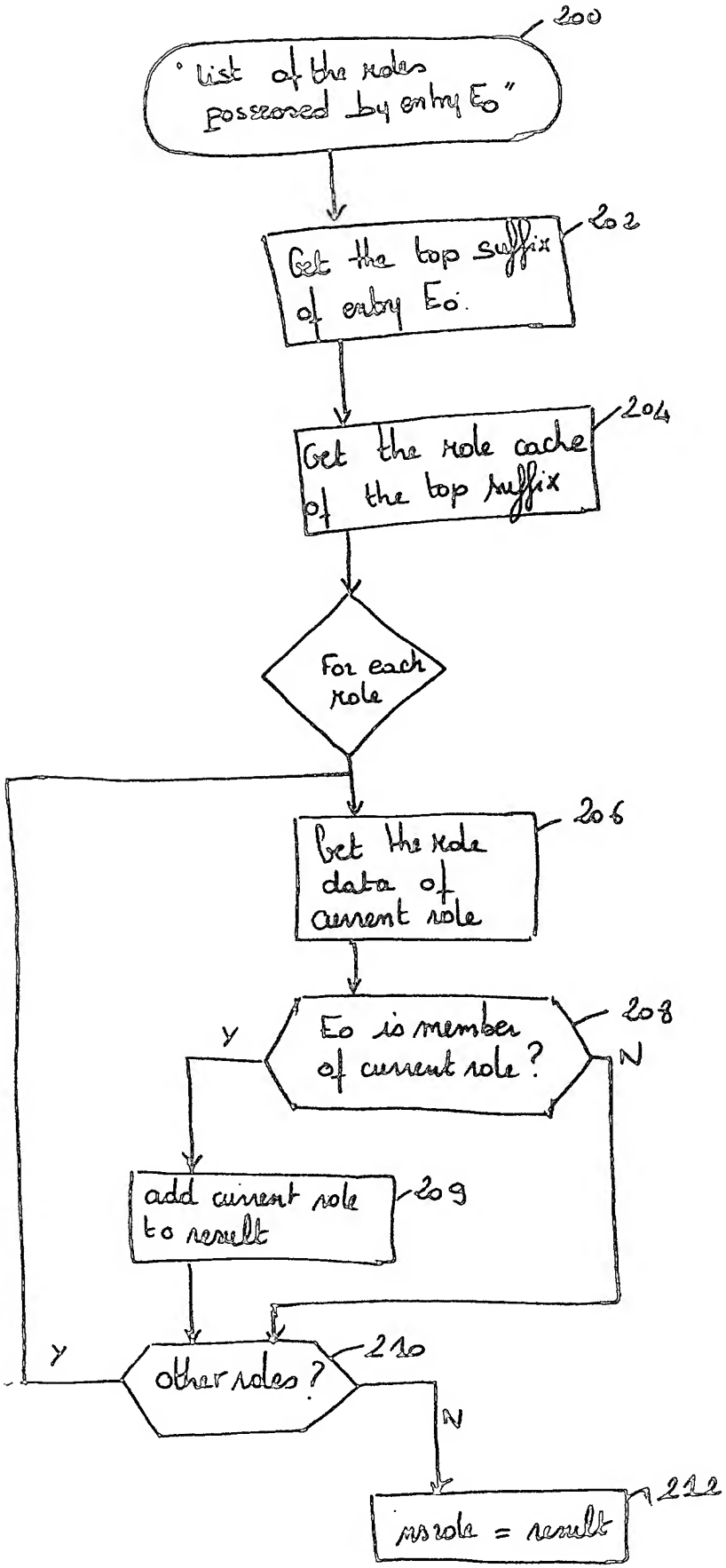


Figure 12



(prior art)

FIG 13a

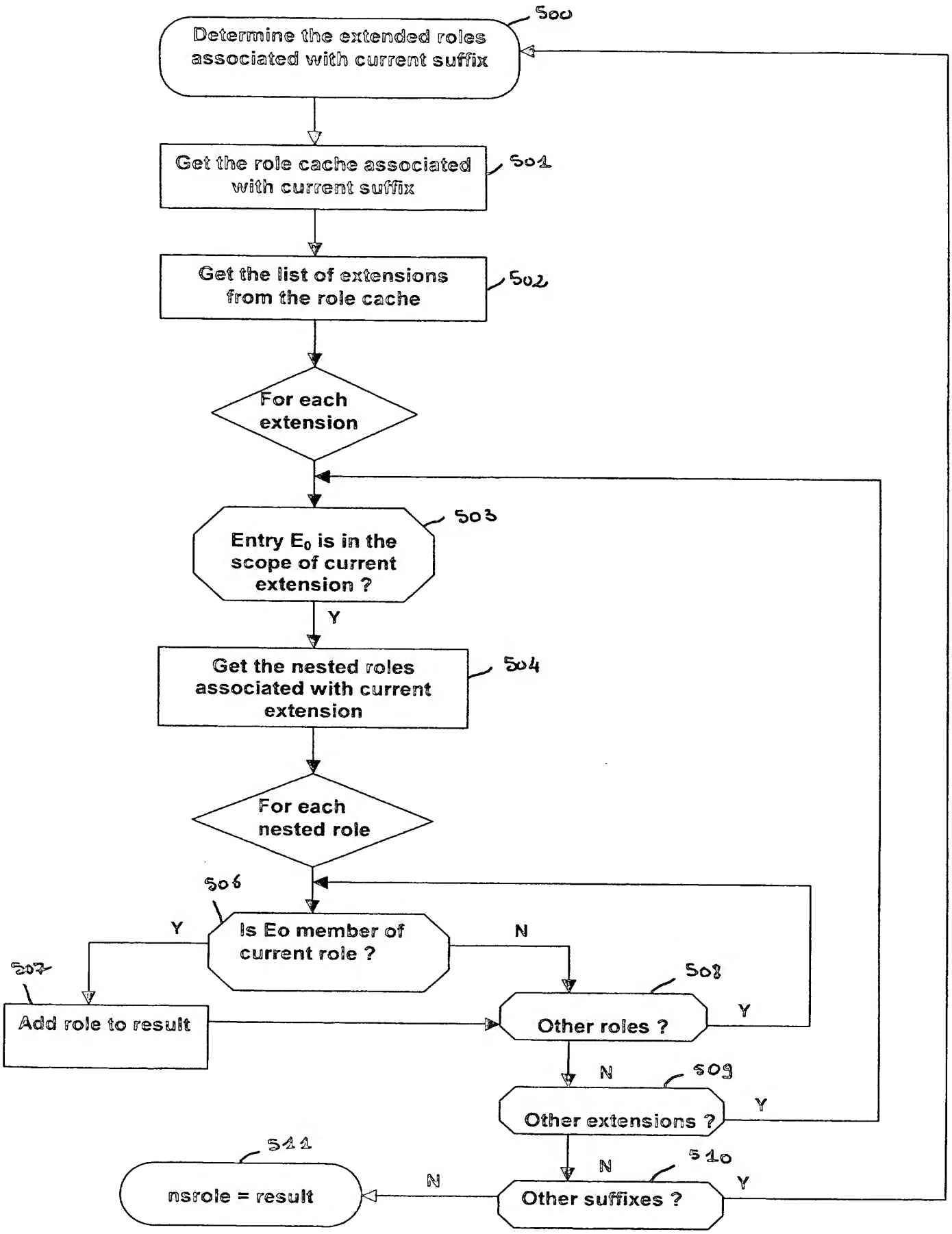

500
Determine the extended roles associated with current suffix
501
Get the role cache associated with current suffix
502
Get the list of extensions from the role cache
For each extension
503
Entry E_0 is in the scope of current extension ?
Y
504
Get the nested roles associated with current extension
For each nested role
506
Is Eo member of current role ?
Y
N
507
Add role to result
508
Other roles ?
Y
N
509
Other extensions ?
Y
N
510
Other suffixes ?
Y
N
511
nsrole = result

Figure 13b

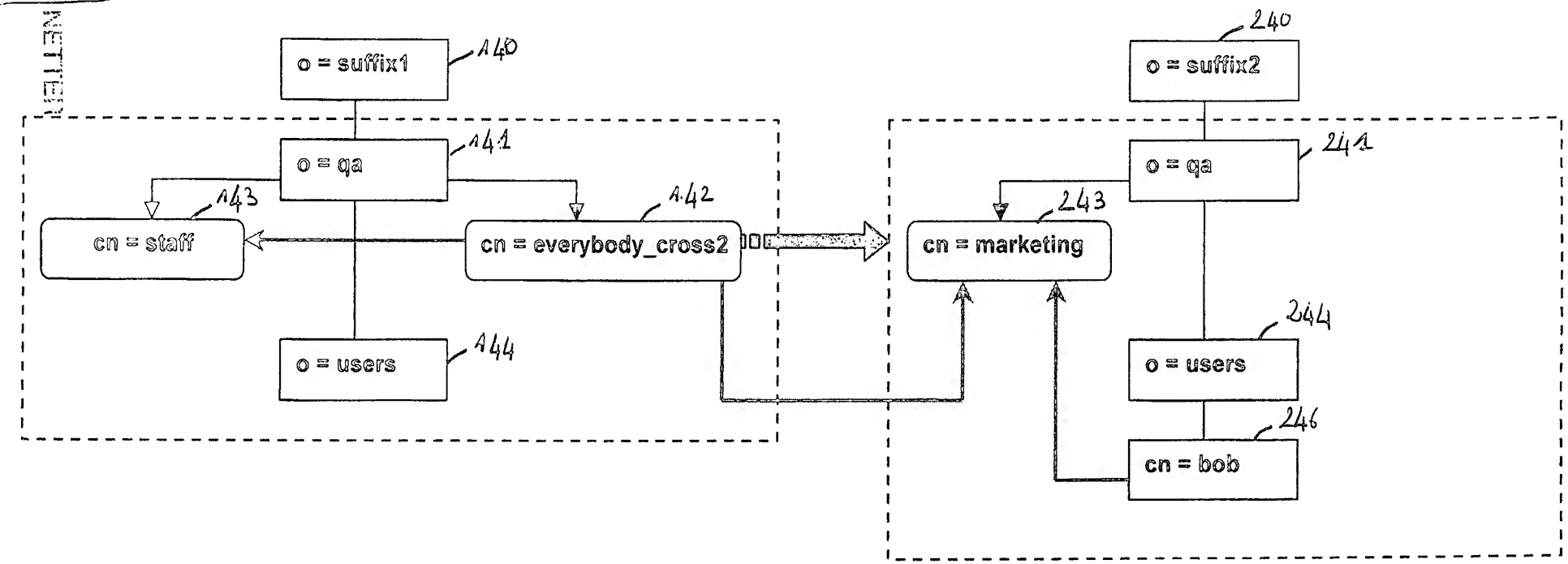

o = suffix1
140
o = qa
141
cn = staff
143
cn = everybody_cross2
142
o = users
144
o = suffix2
240
o = qa
241
cn = marketing
243
o = users
244
cn = bob
246

FIG. 14

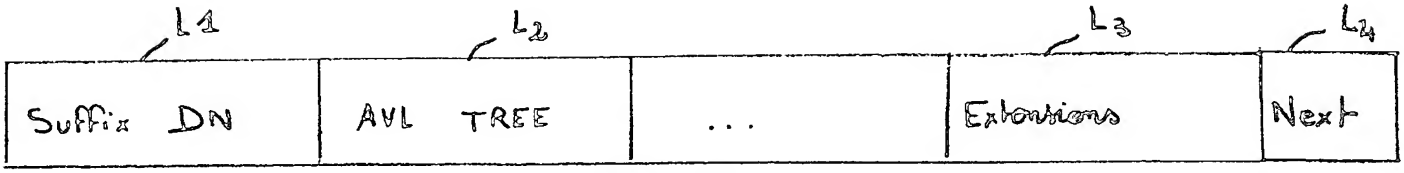


FIG. 15.a

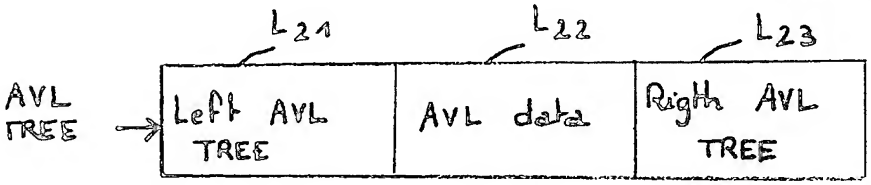


FIG. 15b

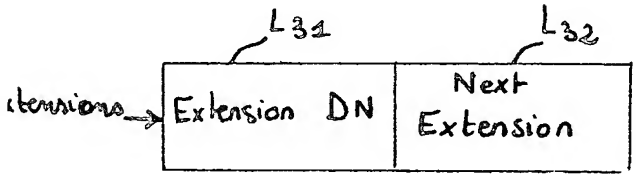


FIG 15 c

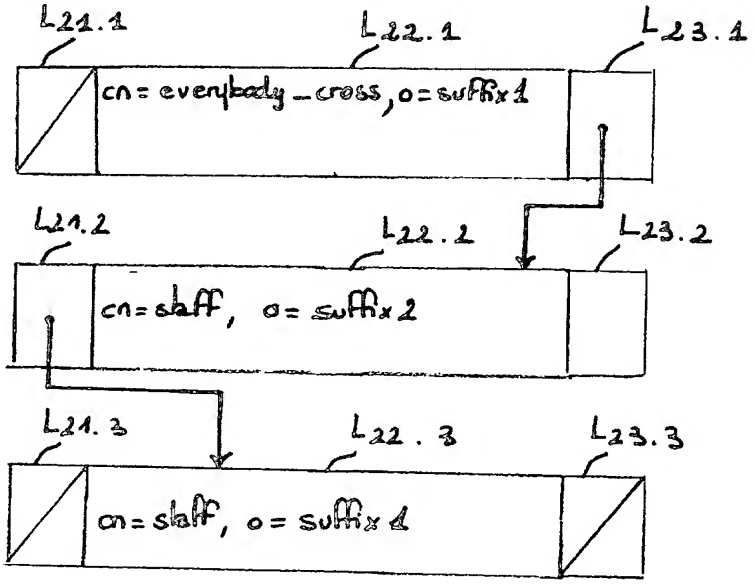


FIG. 16

CABINET NETTER

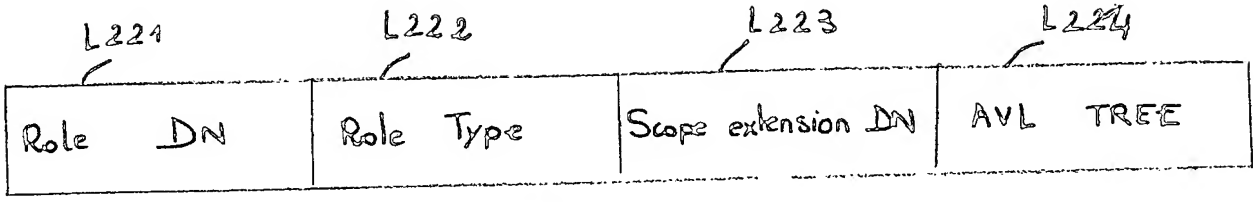


FIG. 17

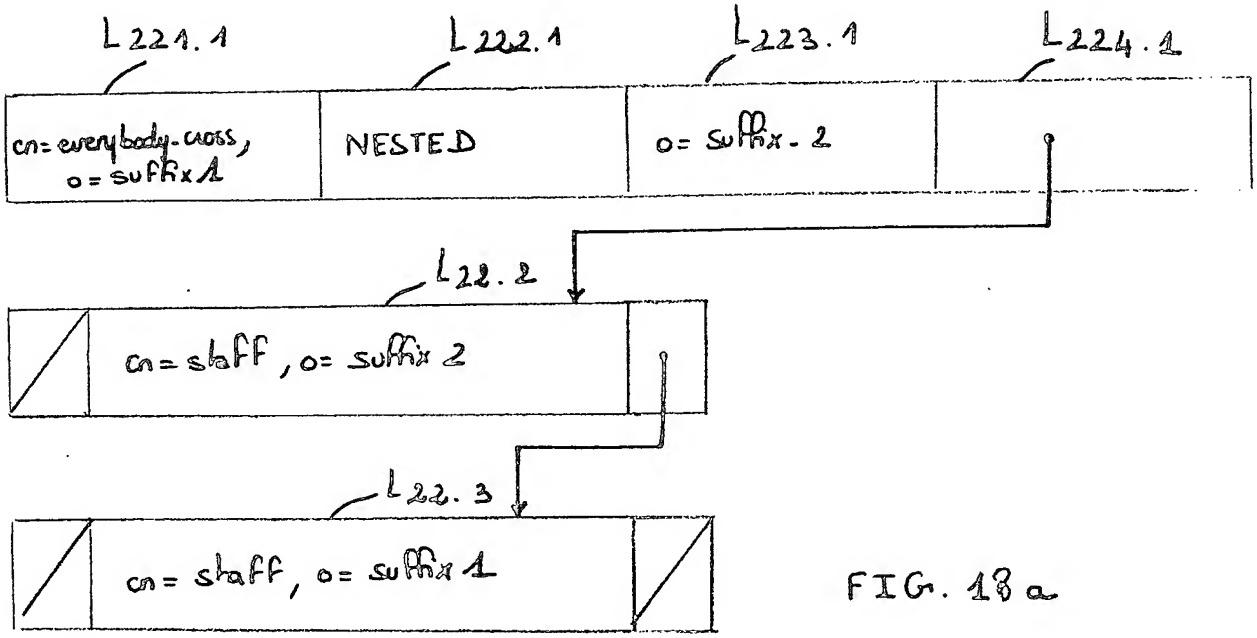


FIG. 18a

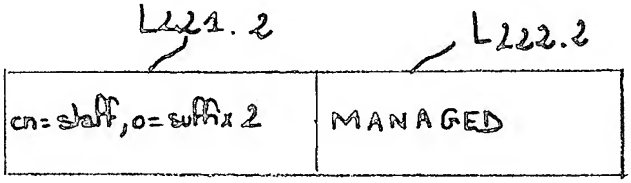


FIG. 18b

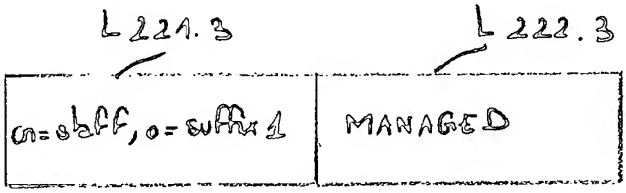


FIG. 18c

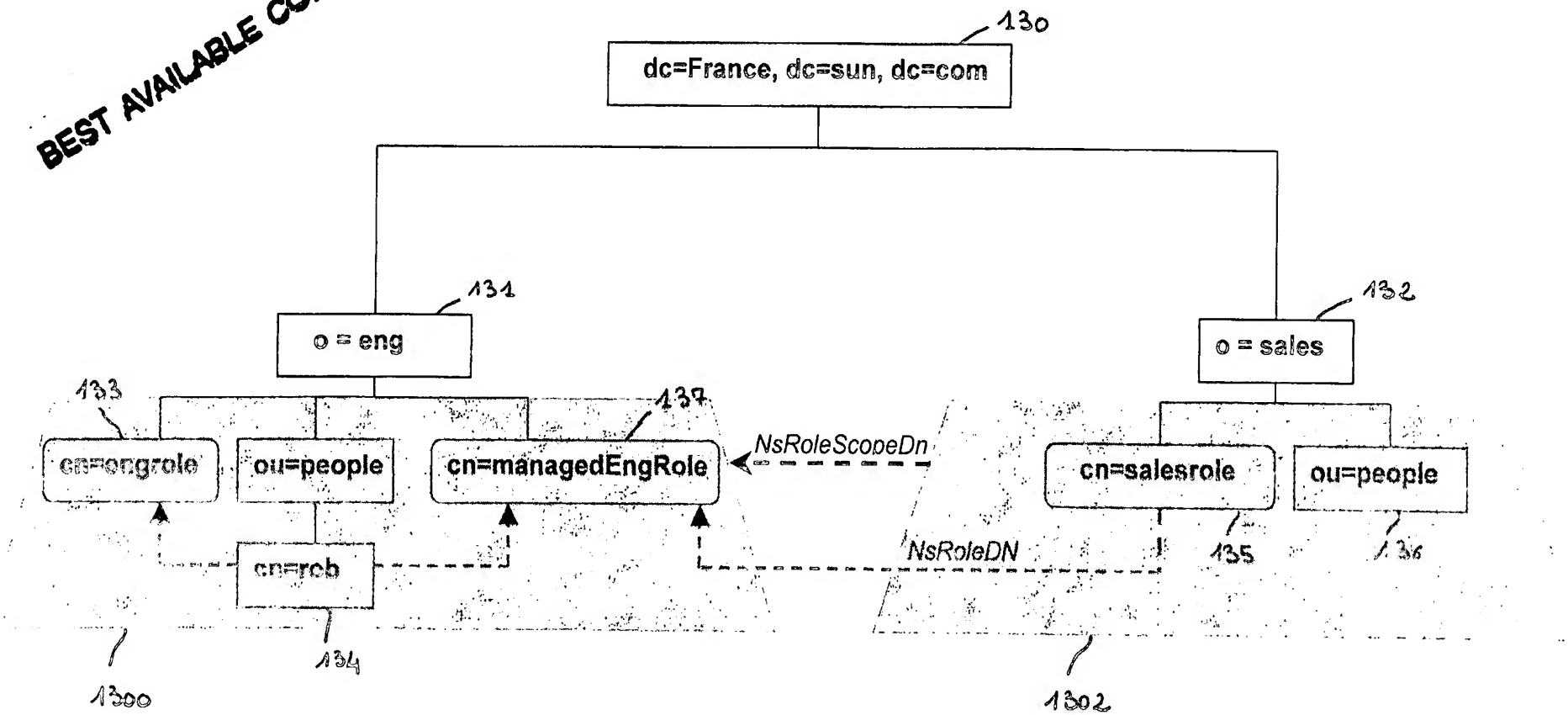


Figure 19